Time Series Models

Taewan Kim

July 14, 2022

Contents

Before jumping into time series notes, I want to state that I used the followings as the main resources for the notes: cshalizi from Carnegie Mellon University and rnau from Duke University. Please take your time to look at them if you want to dive deeper following the link.

1 Exploratory Data Analysis

Data observations are assumed to be independent of each other generally when we analyze data. However, data observations may not be always independent of each other. One of the examples is time series data, of which the past values might affect the current or future values.

1.1 Data Property

1.1.1 Stationarity

Recall that we assume that the data are independent and identically distributed (IID). That is analogous to the property called **stationarity** in time series data. A random variable is called **stationary** if its statistical properties are unchanged over time. That is the variable does not have trend, and its variations around its mean would have a constant amplitude with consistent fashion of wiggling. A random variable of stationarity is regarded as a combination of signal and noise where signal would be a sinusoidal oscillation, possibly containing seasonal components.

1.1.2 Trends

Unfortunately, not all the time series data are stationary. For example, if we see a stock price graph, the plot does not illustrate that the price does not fluctuate around a constant value of mean. Rather, it shows a long-term increase or decrease. Such inclination to increase/decrease is called "trend" hidden in time series data. To tackle this, we want to separate the data into two parts: persistent trend and stationary deviations around the trend.

$$Y_t = X_t + Z_t$$
$$\text{series} = \text{fluctuations} + \text{trend}$$

where

$$Z_t = \mathbb{E}[Y_{\cdot,t}] \approx \sum_{i=1}^{m} Y_{i,t}$$

1.1.3 Seasonality

Not only trend but also seasonality can exist in time series data. Seasonality are the cyclic variations that repeat regularly, which could be daily, weekly, monthly, and etc. The decomposition would look like this.

$$Y_t = X_t + Z_t + S_t$$
$$\text{series} = \text{fluctuations} + \text{trend} + \text{seasonality}$$

where T is the period of cycle, $m = n/T$ is the number of full cycles, and

$$S_t \approx \frac{1}{m} \sum_{j=0}^{m-1} Y_{t+jT}$$

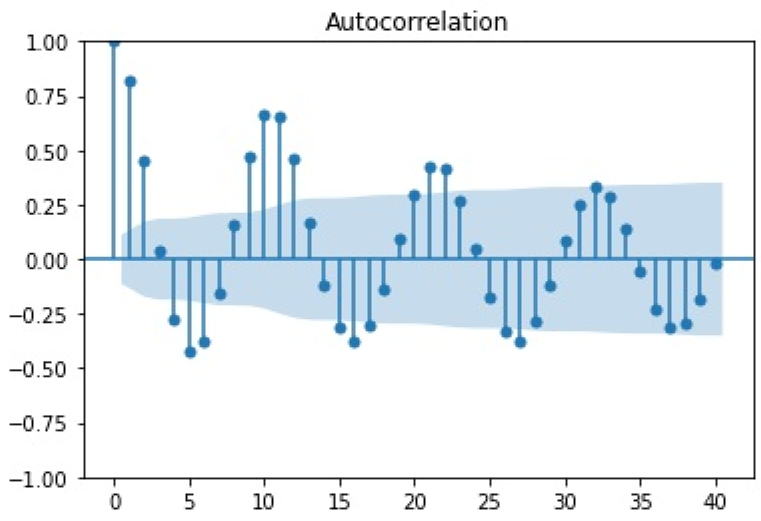

Figure 1: Toy Example of Autocorrelation Plot

1.2 Autocorrelation

Time series are serially dependent that the current value is generally dependent on the previous values. Autocorrelation, often called as serial correlation, is the correlation between the variable and its previous time step values, to check if the time series variable has constant amplitude and wiggling fashion.

$$\rho(h) = \frac{Cov[X_t, X_{t+h}]}{Var[X_t]} = \frac{\gamma(h)}{\gamma(0)} \tag{1}$$

The figure 1 is an example of autocorrelation plot drawn with the data and function provided by statsmodels package. The figure illustrates autocorrelations at different lag values. statsmodels package uses Bartlett's correction to draw the confidence interval of each autocorrelation, which is the shaded area in the figure. For more information about Bartlett's test, refer to this website. Using the confidence intervals, we can identify whether an autocorrelation is statistically significant or not. Correlation values outside of the shaded area are very likely to be meaningful correlation. As an additional note, we can observe that the correlation of x_t and x_{t+h} is decreasing as $h \to \infty$. This is called **correlation decay**. Here are the packages that provide functions to draw autocorrelation plots.

1. Pandas
2. matplotlib
3. statsmodels

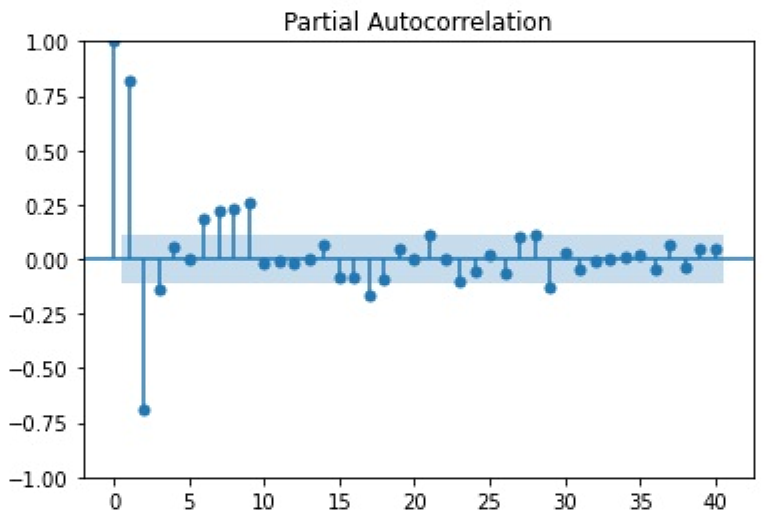

Figure 2: Toy Example of Partial Autocorrelation Plot

1.3 Partial Autocorrelation

Along with autocorrelation, partial autocorrelation is another indispensable characteristic to analyze before diving into modeling step. The autocorrelations might capture direct and indirect correlations, so partial autocorrelation tries to eradicate the indirect correlation part by using the shorter lagged values (controlling the other lags). By looking at the partial autocorrelation, it would give us the important intuition on how to choose the value of p in ARIMA model.

The figure 2 illustrates the plot of partial autocorrelation using the same data from the autocorrelation plot. The way to understand the partial autocorrelation plot is pretty similar to autocorrelation. If the correlation values are located outside of the shaded area, they are highly likely to be statistically significant. From the toy example, we can see that the first four seem to be meaningful.

The package to provide partial autocorrelation plot function is statsmodels

2 Classic/Baseline Models

2.1 ARIMA

ARIMA models are considered as the general models for analyzing a time series data by transforming and/or differencing the data to be stationary. Assuming that the data is stationarized, ARIMA model tackles such time series data **to separate the signal from the noise and make use of the signal to make predictions.**
The ARIMA forecasting equation is linear with the lags of dependent variables and lags of the forecast errors. More specific, if we only use lagged values of Y, it is a pure autoregressive model. However, if we use lags of the errors, then an ARIMA model is not a linear regression model because the errors are computed on a period-to-period basis without any specification on the last period's error.

ARIMA stands for Auto-Regressive Integrated Moving Average.

1. Auto-Regressive: lags of the stationarized random variables are called "autoregressive" terms
2. Integrated: time series data to be made stationary by differencing
3. Moving Average: lags of the forecast errors

Corollary, ARIMA models have three parameters.

1. p, the number of autoregressive terms
2. d, the number of nonseasonal differences to fulfill stationarity
3. q, the number of lagged forecast errors

Mathematically, ARIMA can be described as follows

$$\widehat{y}_t = \mu + \underbrace{\sum_{i=1}^{p} \phi_i y_{t-i}}_{\text{AR(p)}} + \underbrace{\sum_{i=1}^{q} \theta_i \epsilon_{t-i}}_{\text{MA(q)}} + \epsilon_t \tag{2}$$

where ϕ and θ are the parameters for the Auto-regressive and Moving Average parts, respectively. ϵ_i are the error terms, which are assumed to be i.i.d from normal distribution with mean of 0.

Now, we are going to look at a number of special cases of ARIMA models.

2.1.1 First-order autoregressive model

First-order autoregressive model is denoted as ARIMA(1, 0, 0). Intuitively, if the time series is stationary and autocorrelated, it predicts future values, using a sequence of its previous values. This special case uses the lagged value by one period as the notation implies. The equation is as follows.

Mathematically,

$$\widehat{Y}_t = \mu + \phi_1 Y_{t-1} \tag{3}$$

Note that the constant term would be excluded if the mean of Y is zero.

2.1.2 Random Walk

Random walk model is denoted as ARIMA(0, 1, 0). This simplest model can be applied if the time series variable is not stationary.

Mathematically,

$$\widehat{Y}_t - Y_{t-1} = \mu \longrightarrow \widehat{Y}_t = \mu + Y_{t-1} \tag{4}$$

The constant term is the average of periodic changes in Y.

2.1.3 Differenced first-order autoregressive model

If the errors from a random walk model are autocorrelated, then the performance of the model would be poor. In order to tackle this issue, we could add one lag of the variable in the prediction model. This model idenoted as ARIMA(1, 1, 0)

Mathematically,

$$\widehat{Y}_t - Y_{t-1} = \mu + \phi_1(Y_{t-1} - Y_{t-2}) \longrightarrow \widehat{Y}_t = \mu + Y_{t-1} + \phi_1(Y_{t-1} - Y_{t-2}) \tag{5}$$

2.1.4 Simple exponential smoothing

This model is also suggested to tackle the problem that we encounter when the errors from a random walk model are auto-correlated. If a random variable is non-stationary (i.e., showing fluctuations around a varying mean), the random walk model and moving average of pasts values are likely to fail to explain the data. In such case, it could be better to use an average of the previous few observations in order to filter out the noise. That is what exponential smoothing does. Note that this model is denoted as ARIMA(0, 1, 1) without constant, meaning that it does not have a constant term.

Mathematically,

$$\begin{aligned}\widehat{Y}_t &= \widehat{Y}_{t-1} + \alpha e_{t-1} \\ &= Y_{t-1} - (1-\alpha)e_{t-1}\end{aligned}$$

where $e_{t-1} = Y_{t-1} - \widehat{Y}_{t-1}$ by definition. Then,

$$\widehat{Y}_t = Y_{t-1} - \theta_1 e_{t-1} \tag{6}$$

This is the ARIMA(0, 1, 1) - without constant where $\theta_1 = 1 - \alpha$

Remark It is more effective to add an AR term to the model when autocorrelation is positive. If it is negative, MA terms would improve the performance.

2.1.5 Simple exponential smoothing with growth

This model is the simple exponential smoothing with constant term, denoted as ARIMA(0, 1, 1) with constant. By including a constant term in the model, we could estimate an average non-zero trend.

Mathematically,

$$\widehat{Y}_t = \mu + Y_{t-1} - \theta_1 e_{t-1} \tag{7}$$

2.1.6 Linear exponential smoothing

These models use two nonseasonal differences with MA terms. Note that the second difference of Y is not the difference between Y and its lagged variable by two periods. The second difference of Y is as follows

$$(Y_t - Y_{t-1}) - (Y_{t-1} - Y_{t-2}) = Y_t - 2Y_{t-1} + Y_{t-2}$$

. This second difference is analogous to a second derivative of a continuous function that measures the acceleration at a given point. This model is denoted at ARIMA (0, 2, 1) without constant.
Additionally, The ARIMA (0, 2, 2) model without constant predicts that the second difference equals a linear function of the last two forecast errors.

Mathematically,

$$\widehat{Y}_t - 2Y_{t-1} + Y_{t-2} = -\theta_1 e_{t-1} - \theta_2 e_{t-2}$$

Rearranging,

$$\widehat{Y_t} = 2Y_{t-1} - Y_{t-2} - \theta_1 e_{t-1} - \theta_2 e_{t-2} \tag{8}$$

where θ_1 and θ_2 are the MA(1) and MA(2) coefficients. Using exponentially weighted moving averages, it estimates local level and local trend in the variable.

2.1.7 Damped-trend linear exponential smoothing

This model is denoted as ARIMA(1, 1, 2) without constant.

Mathematically,

$$\widehat{Y_t} = Y_{t-1} + \phi_1(Y_{t-1} - Y_{t-2}) - \theta_1 e_{t-1} - \theta_2 e_{t-2} \tag{9}$$

The models are generally reliable when at least one of p and q is not larger than 1. (A model like ARIMA(2,1,2) may suffer from over-fitting) Further resource on damped-trend linear exponential smoothing can be found **here**

2.2 PROPHET

Facebook, or now Meta, published an article called Forecasting at Scale (Taylor and Letham, 2017). The paper proposes a new model called Prophet to produce forecasts at scale, for a few different perspectives of notions. The scales are

1. Numerous people can make forecasts, without training in time series methods
2. Various problems with potentially idiosyncratic features can be solved
3. A large number of predictions are created, necessitating efficient and automated means of evaluating and comparing them, and detecting when models perform poorly.

Prophet is a decomposable time series model with three components: trend, seasonality, and holidays.

$$y(t) = g(t) + s(t) + h(t) + \epsilon_t \tag{10}$$

where $g(t)$ is the trend function for non-periodic changes, $s(t)$ illustrates periodic changes, and $h(t)$ is the effect function of holidays. ϵ_t is the error term, assumed to be normally distributed.

The model is similar to a generalized additive model (GAM), but Prophet only uses time as a regressor, possibly with several linear and non-linear functions of time as components. Prophet has the following advantages:

1. Easily accommodate seasonality with multiple periods and analysts are able to make different assumptions
2. The measurements do not have to be spaced regularly, implying that missing values do not have to be imputed.
3. Easily interpret-able and changeable parameters allow analysts to impose assumptions on the forecast.

2.2.1 Trend Model

Prophet implements two trend models: saturating growth model and piece-wise linear model.

- Nonlinear, Saturating growth Generally, the growth is modeled by the logistic growth model,

  $$g(t) = \frac{C}{1 + exp(-k(t-m))}$$

  where C, the carrying capacity, k, the growth rate, and m, an offset parameter.

- Prophet assumes that the carrying capacity and the growth rate are not constant, as events such as new product introduction may change the rate of growth.
- Prophet, therefore, takes such changepoints to allow the growth rate to be changed.
- If we have S changepoints at time $s_j, j = 1, ..., S$, $\boldsymbol{\delta} \in \mathtt{R}^S$ is a vector of rate adjustments where δ_j is the change in rate at time s_j.
- The rate at any time t is $k + \sum_{j:t>s} \gamma_j$ where k is the base rate. More clearly with defining additional vector $\mathbf{a}(t) \in 0, 1^S$. Then, the rate is $k + \mathbf{a}(t)^T \boldsymbol{\delta}$.

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq s_j \\ 0, & \text{otherwise} \end{cases}$$

- If we adjust the rate k, then the offset parameter m should be also adjusted to connect the endpoints of the segments. Then, the correct adjustment γ_j at changepoint j is computed as

$$\gamma_j = (s_j - m - \sum_{l<j} \gamma_l)(1 - \frac{k + \sum_{l<j} \delta_l}{k + \sum_{l \leq j} \delta_l}$$

- Finally, the piece-wise logistic growth model is proposed as

$$g(t) = \frac{C(t)}{1 + exp(-(k + \mathbf{a}(t)^T \boldsymbol{\delta})(t - (m + \mathbf{a}(t)^T \boldsymbol{\gamma}))} \tag{11}$$

  where $C(t)$ is the expected capacities of the system at any point in time.

- Linear Trend with Changepoints
  - A piece-wise constant rate of growth provides often useful model if no saturating growth is prevalent in the data.
  - The trend model is

$$g(t) = (k + \mathbf{a}(t)^T \boldsymbol{\delta})t + (m + \mathbf{a}(t)^T \boldsymbol{\gamma}) \tag{12}$$

  where k is the growth rate, $\boldsymbol{\delta}$ has the rate adjustments, m is the offset parameter, and γ_j is $-s_j \delta_j$ to make it continuous.

2.2.2 Seasonality

- It utilizes Fourier series to provide a flexible model of periodic effects. The approximate arbitrary smooth seasonal effects with the Fourier series is

$$s(t) = \sum_{n=1}^{N} (a_n \text{cos}(\frac{2\pi nt}{P}) + b_n \text{sin}(\frac{2\pi nt}{P}))$$

  where P is the regular period of the time series variable (ex: P = 365 for yearly data or P = 7 for weekly data)

- Further, a matrix of seasonality vectors for each t is constructed to represent the seasonality component.

$$X(t) = [cos(\frac{2\pi(1)t}{P}), ..., sin(\frac{2\pi(10)t}{P})] \tag{13}$$

  if $N = 10$ and it is yearly seasonality.

- Finally, the seasonal component is

$$s(t) = X(t)\boldsymbol{\beta}. \tag{14}$$

- Increasing the value of N will work well on the data with seasonal patterns that change more quickly.

2.2.3 Holidays and Events

Holidays and occasions often show somewhat expected fluctuation to time series data that they may not follow the distribution of the data, possibly regarded as outliers. For example, the volume of shopping will significantly increase in the Thanksgiving day week. Therefore, it is important for us to incorporate this information when predicting with a model. Prophet allows a list of past and future holidays to be fed into the model so that the model take account of those days, with assumption that the effects of holidays are independent.

2.2.4 Model

prophet allows users to alter the model by changing the values of parameters.

1. Capacities: Users can specify capacities.
2. Changepoints: Users can specify the number of changepoints
3. Holidays and seasonality: Users can provide the holiday and event dates and applicable time scales of seasonality.
4. Smoothing parameters: Users can adjust τ to alter the model to be more global or locally smooth models (trend flexibility). Also, additional parameters (σ, ν) allow the users to tell the model about the expected seasonal variation, accordingly.

The Prophet, therefore, is able to predict weekly and yearly seasonalities, without overreacting to the holiday changes. Because the Prophet is a decomposable model, we can analyze more deeply by looking at the trend, weekly seasonality, and yearly seasonality components.

More detailed description and API can be found this article and API website

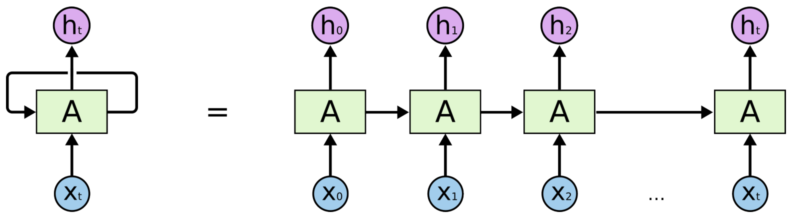

Figure 3: Recurrent Neural Network from (Olah, 2015)

3 Deep Learning Model

People think before speaking or acting something. Based on the previous experiences and understanding, we think which action or choose of words might be more effective for our ends. That is, we are persistent that we do not forget the information and memories but recall them when necessary. The classic neural network models, such as deep neural networks, do not replicate this characteristic of humans. For time series data, it is important for models to use the previous data or events and to make better forecasts. To tackle this issue, a number of different deep learning models are introduced.

3.1 RNN

Recurrent Neural Networks (RNN) has loops inside the network that other classic neutral networks did not have. By looping itself, the previous information becomes persistent over predictions (or time in our case).

The figure 3 illustrates how a RNN is running its job. As you can see the illustration, the recurrent neural network has the chain shaped architecture that the previous prediction actually affects the current and future predictions.

Although RNN seems to be a perfect neural network model as it is persistent, it faces some issues.

- Long-term Dependency: In practice, RNN would not be able to use meaning information from the past if the information is far away from the current data point (or current time)
- Optimization: RNN might suffer from exploding/vanishing gradient problems, if the sequence is long.

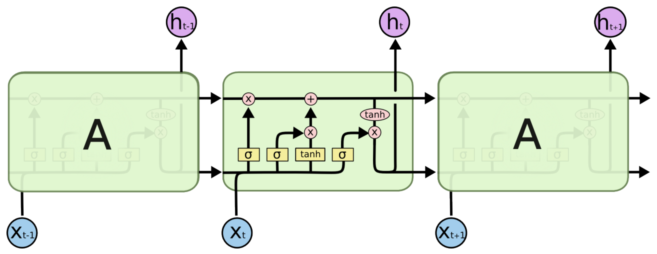

Figure 4: Long Short Term Memory Neural Network from (Olah, 2015)

3.2 LSTM

Long Short Term Memory networks (LSTM) are a subspecies of RNN, addressing the issue of RNN. Unlike RNN, LSTM is able to capture the long-term dependencies, by removing or adding information to the next cell state.

The figure 4 depicts the architecture of LSTM model. There are four important components in the LSTM architecture.

1. Cell state, the top line of the figure, is the key of LSTM that it passes down toward the next prediction with some modifications. Its input is C_{t-1} and output C_t with changes.
2. Forget Gate takes h_{t-1} and x_t and outputs a number within the range of 0 and 1 for each number in C_{t-1} to represent whether it keeps or erases the previous information. (0 means completely disregards.)
3. Input Gate takes C_t and h_{t-1} and determines what new information will be stored in that current cell state.
4. Output Gate takes h_{t-1}, x_t, and C_t and controls how much information will be transmitted to the hidden vector.

LSTM can avoid vanishing gradients because of the forget gate where the range stays between 0 and 1 to get a better control of gradient values. Also, the cell state gradient is an additive function.

4 Model Evaluation

4.1 Regression

Consider a model, f, that estimates the relationship between dependent variables, X and independent variable Y so that it predicts a continuous value.

$$Y_i = f(X_i, \beta) + e_i$$

4.1.1 Mean of Absolute Error (MAE)

Mean of Absolute Error (MAE) is a measure of errors between observations and predicted values. It is commonly used in time series analysis because it gives how close the model's predictions are to true values.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - \widehat{y}_i|}{n} = \frac{\sum_{i=1}^{n} |\text{err}_i|}{n} \tag{15}$$

4.1.2 Mean Squared Error (MSE)

Mean Squared Error (MSE) is also commonly used for a performance metric on regression models. If we take an observed dataset to compute MSE, it is often called "empirical risk", the estimate of the true MSE. MSE gives more weight to larger errors, sensitive to outliers. As a note, MSE is also often used for calculating the variance and/or bias of estimators, $\text{MSE}(\widehat{\theta}) = \text{var}(\widehat{\theta}) + \text{Bias}^2(\widehat{\theta})$.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\text{err}_i)^2 \tag{16}$$

Additionally, we often use root-mean-square error (RMSE) as a measure of errors. In some cases, MSE might have a large integer, so taking root of MSE would give us a better measure to compare the performances of models. It is also a standard deviation of an unbiased estimator.

$$\text{RMSE} = \sqrt{\text{MSE}^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y}_i)^2} \tag{17}$$

4.1.3 Mean Absolute Percentage Error (MAPE)

Mean absolute percentage error (MAPE) is a relative measure of errors in regression models. MAPE is effective for larger sets of data. However, it is almost impossible to apply it to datasets that contain (nearly) zero (division by zero and exploding issue). Contrast to the above two measures, it expresses the prediction accuracy as a ratio. Also, MAPE is easy to interpret that the average deviation between the predicted values and true values are 5%, if the MAPE is 5%.

$$\text{MAPE} = \frac{1}{n}\sum_{t=1}^{n} \left|\frac{A_t - \widehat{Y}_t}{A_t}\right| \tag{18}$$

where A_t is the actual value and $\widehat{Y}_t$ is the predicted value.

4.2 Classification

A classification model identifies which categories an observation belong to.

Consider a binary classification model, f, that $y = f(X, \beta) = \beta_0 + \beta_1 x$.

$$p(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}} \tag{19}$$

If $p(x) > 0.5$, the model will classify it as 1 (positive class). Otherwise, the model will classify as 0, negative class.

For a multi-class classification model in perspective of GLM,

$$\frac{p_{ik}}{p_{ik} + p_{ic}} = F(X_i^T \beta_k)$$

where c is an arbitrary baseline category and F is a link function such as logit, $\text{logit}(p) = \log(\frac{p}{1-p})$. Further, if F is a logit link,

$$\frac{p_{ik}}{p_{ic}} = e^{X_i^T \beta_k}$$

. Using the baseline-category logit model, we get

$$p_{ik} = \frac{e^{X_i^T \beta_k}}{1 + \sum_{h=1}^{c-1} e^{X_i^T \beta_h}} \tag{20}$$

4.2.1 Confusion Matrix

If you are a statistician, data scientist, machine learning engineer, etc., you must be familiar with confusion matrix. Confusion matrix is a great tool to see the performance of a model clearly. Using the confusion matrix, we can compute a number of different metrics, such as precision, recall, specificity, etc.

Using the above table, now we can measure any metric regarding classification performance.

		Predict	
	Total P + N	Positive (PP)	Negative (PN)
Actual	Positive (P)	True Positive (TP)	False Negative (FN)
	Negative (N)	False Positive (FP)	True Negative (TN)

4.2.2 Precision

Precision is a measure of performance in classification task. It is often called ”positive predictive value”. Precision measures how the model correctly classifies observations as positive among the positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{PP}} = 1 - \text{FDR} \tag{21}$$

4.2.3 Recall

Recall, often called ”sensitivity” and ”True Positive Rate”, is another measure of classification measure that it measures how well the model classifies observations as positive out of all positive observations. As a note, true negative rate is called ”specificity”.

$$\text{Recall} = \frac{\text{TP}}{\text{P}} = 1 - \text{FNR} \tag{22}$$

where FNR stands for False Negative Rate

Note that recall and precision have a trade-off relationship if a threshold for classification is changed.

4.2.4 F-1 Score

F-1 score is a measure of accuracy on test dataset by the harmonic mean of precision and recall metrics. The highest possible value is 1, indicating perfect precision and recall.

$$F_1 = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{\text{TP}}{\text{TP} + \frac{1}{2}(\text{FP} + \text{FN})} \tag{23}$$

References

Olah, C. (2015). Understanding lstm networks.

Taylor, S. J. and Letham, B. (2017). Forecasting at scale. *Peer J Preprints*.